Graph Process Specifications for Hybrid Networked Systems ⋆

Philip Twu * **Patrick Martin** * **Magnus Egerstedt** *

* *Department of Electrical and Computer Engineering*
Georgia Institute of Technology, Atlanta, GA 30332
(e-mail: {ptwu,patrick.martin}@gatech.edu, magnus@ece.gatech.edu)

Abstract: Research in multi-agent systems has supplied a diverse collection of decentralized controllers to accomplish specific tasks. When agents execute a sequence of these controllers, the network behaves as a hybrid system, where the dynamics in each mode evolve according to a single controller in the sequence. This paper presents a formal specification for such a system that describes the underlying graph process associated with the information flow amongst agents in each mode. Since many decentralized controllers require specific information graph topologies in order to function properly, a problem that arises is that the information graph at the termination of one mode may not be sufficient to initiate the next mode in the sequence. We propose a Graph Process Specification (GPS) framework that describes the graph process. Furthermore, if two modes cannot be executed consecutively, a GPS provides a way to determine which modes can be inserted in between them to make the resulting sequence executable. We formally define a GPS, describe its execution, and provide examples that showcase its usage in composing together multiple decentralized controllers within a multi-agent system.

Keywords: Decentralized control, formal specification, graph theoretic models, hybrid modes, network topologies.

1. INTRODUCTION

Research in multi-agent systems has supplied a wide variety of decentralized controllers that are each designed to achieve particular objectives. Many large-scale multi-agent missions can be decomposed into a sequence of small tasks. In order to achieve the final objective of the mission, the agents must execute a sequence of controllers, where each controller is responsible for accomplishing one of the intermediate tasks.

Many controllers, however, require a certain minimum information flow amongst the agents for their operation. For example, agents executing a nearest-neighbor averaging rule (e.g. Olfati-Saber et al. (2007)) require that the information graph topology be a connected graph in order for all the agents to reach consensus. What we want to do is specify a sufficiently rich initial topology and obtain guarantees that this initial specification ensures the proper operation of the controller.

One potential obstruction to focusing on the initial topology is that the graph topology may change with time as the agent states evolve. Moreover, it is not always possible to have the agents in the network consecutively execute a pair of arbitrarily chosen controllers. This situation occurs because there is no guarantee that the information graph resulting from the execution of the first controller satisfies the minimal information flow requirements to start executing the second controller. To make these observations concrete and to characterize when such controllers can be concatenated, we propose a framework in which the evolution of the system (a so-called graph process as discussed in Mesbahi and Egerstedt (2010)) can be formally described. We call this framework a Graph Process Specification (GPS).

Given a sequence of controllers to be executed, a GPS describes how the information graph evolves with the application of each of the controllers. More specifically, for each controller in the sequence, it describes the set of minimum information graphs that are required to initiate the controller and the set of final information graphs guaranteed at the termination of the controller. With this information, it is possible to verify whether the agents in the system can execute a specified sequence of controllers. This verification is performed by first checking that the initial graph topology can initiate the first controller in the sequence. Furthermore, for each pair of controllers that are to be executed consecutively in the sequence, a check needs to be performed to ensure that the resulting information graph from the application of the first controller meets the minimal requirements to initiate the second controller.

In addition to checking whether a given sequence of controllers can be executed by the agents in the system, a GPS can also be used to determine how to alleviate the problem when the check fails. Suppose there is a pair of controllers that cannot be executed consecutively because the first controller does not terminate with an information graph rich enough to initiate the second controller. By "richness" we mean that the edge set of the information graph must at least have the edges needed to start the next controller.

⋆ This work was sponsored by the US National Science Foundation through Grant # CCF 0820004.

To alleviate this problem, an additional sequence of controllers, which can be executed consecutively, is inserted between the pair (corresponding to an insertion of modes in the hybrid system) to act as "glue". When performing this gluing procedure, the inserted sequence must be able to start executing with the information graph resulting from the termination of the first controller in the original pair. In addition, the inserted sequence must also terminate with an information graph rich enough to initiate the second controller in the pair.

The work presented in this paper is related to recent developments in abstraction-based approaches to controlling groups of agents. Recent developments in embedded graph grammars (EGG) (e.g. McNew and Klavins (2006); Smith et al. (2009)) develops rules for a collection of agents to choose their local controllers. When a rule fires, the agents switch to the appropriate controller specified in the EGG language. Furthermore, the work in Kloetzer and Belta (2007) used linear temporal logics to specify control for a team of agents. Using this approach, the multi-agent "program" is checked for correctness before individual control laws are issued to the systems.

Since a GPS specifies a sequence of decentralized controllers, it is more closely related to motion description languages (MDL) Brockett (1988); Manikonda et al. (1998); Martin et al. (2008). In particular, Martin et al. (2008) created the MDLn framework to allow for multi-agent motion programs with networked information requirements built into the language. GPS differs from this work by considering a group-level view of the agents, rather than the execution of the individual agents' MDLn strings.

This paper will be organized as follows: Section 2 develops the notions required to describe the execution of a GPS for multi-agent systems. In Section 3, we conclude with an example of how to construct a GPS describing agents first going into a line formation, and then transitioning into a circle formation.

2. GRAPH SPECIFICATION FORMULATION

2.1 Networked System Representation

Consider a collection of N agents, where $x^i \in X$ denotes the state of the ith agent, for $i \in \mathcal{N} = \{1, \dots, N\}$. Additionally, let $x \in X^N$ be the concatenated states of all N agents in the system such that $x = [(x^1)^T \dots (x^N)^T]^T$. The information flow amongst agents at each instant can be described using an undirected graph $G = (\mathcal{N}, E)$, where $E \subseteq \mathcal{E} = \{(i,j) \,|\, i,j \in \mathcal{N}\}$ and $(i,j) \in E$ represents a flow of information between agents i and j. We refer to this as the current *information graph* of the network.

We represent the set of *all* possible information graphs as $\mathcal{G} = \{(\mathcal{N}, E) \,|\, E \subseteq \mathcal{E}\}$. Let the mapping $s : X^N \to \mathcal{G}$ be a *graph inducing function* that takes in the states of all agents in the network and returns an information graph describing the current flow of information. Furthermore, have $\mathcal{S} = \{s \,|\, s : X^N \to \mathcal{G}\}$ be the set of all possible graph inducing functions. Note that this formulation is similar to that of connectivity graphs in Muhammad and Egerstedt (2005).

2.2 Atoms and Consistency

We begin by defining an "atom", which describes the graph process resulting from the usage of a single controller. Such an atom must contain a description of the set of graph inducing functions $\mathbb{S}$, agent dynamics $\mathbb{F}$, initial graphs $\mathbb{G}$, final graphs $\mathbb{H}$, a control law $\mathcal{U}$, as well as a terminal condition $\mathcal{C}$. Formally, we define an atom as follows:

Definition 2.1. An *atom* $\mathcal{A}$ is a tuple given by

$$\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C}),$$

such that

(1) $\mathbb{S} \subseteq \mathcal{S}$
(2) $\mathbb{F} \subseteq \{f \in \mathcal{F} \,|\, f : X^N \times U^N \times \mathbb{R}^+ \to (TX)^N\}$
(3) $\mathbb{G} \subseteq \mathcal{G}$
(4) $\mathbb{H} \subseteq \mathcal{G}$
(5) $\mathcal{U} : \mathcal{S} \times X^N \times \mathbb{R}^+ \to U^N$
(6) $\mathcal{C} : \mathcal{S} \times X^N \times \mathbb{R}^+ \to \{0,1\}^N$.

Here, $\mathcal{F}$ is the set of all functions that are Lipschitz continuous in its first two arguments and piecewise continuous in its third argument, U is the set of control inputs, and TX is the tangent space to X.

Suppose the information graph is initially a supergraph of some element in the atom's set of initial graphs. We will call an atom "consistent" if the graph process is guaranteed to evolve such that it enters and remains in the set of final graphs in finite time. This guarantee on the resulting information graph, when using the controller, is used to decide which controllers can be executed immediately after the current controller has terminated.

Before formally defining the consistency of an atom, we must establish some additional notation. For any function $z : A \to B^N$, where A and B are arbitrarily defined sets, let $z^i : A \to B$, for $i \in \mathcal{N}$, be such that $\forall\, a \in A$, $z(a) = [(z^1(a))^T \dots (z^N(a))^T)]^T$.

Definition 2.2. An atom $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$ is *consistent* when $\forall\, y \in X^N$, $\forall\, s \in \mathbb{S}$ such that $\exists\, g \in \mathbb{G}$ where $g \subseteq s(y)$, and $\forall\, f \in \mathbb{F}$, if

(1) $\dot{x}(t) = f(x(t), \mathcal{U}(s, x(t), t), t)$
(2) $x(t_0) = y$ for some $t_0 \in \mathbb{R}^+$,

then $\mathcal{C}^i(s, x(t), t) = 1$, for any $t \geq t_0$ and $i \in \mathcal{N}$, implies that $s(x(t)) \in \mathbb{H}$. Furthermore, $\exists\, t^* \in [t_0, \infty)$ and $\exists\, j \in \mathcal{N}$ such that $\mathcal{C}^j(s, x(t), t) = 1\, \forall\, t \geq t^*$.

The above definition is summarized as follows: given an atom $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$, assume that the N agents have dynamics given by $f \in \mathbb{F}$ and information flow amongst agents determined by the graph inducing function $s \in \mathbb{S}$. Suppose that the graph induced by the initial agent states at time t_0 has an information flow topology that is a supergraph of any of the minimal requirement graphs given in $\mathbb{G}$, and that the controller $\mathcal{U}$ is used. If $\mathcal{A}$ is consistent, then it is guaranteed that the system will evolve such that the information graph enters and stays in the set $\mathbb{H}$ in finite time. Membership of the current information graph in the set $\mathbb{H}$ is indicated by when $\mathcal{C}^i \to 1$, for some agent $i \in \mathcal{N}$. A consistent atom only requires that a single agent realize in finite time when the graph process has entered into and will stay in the set of final graphs.

Note that the definition of a consistent atom does not prevent an agent from computing its control using state information from other agents that it does not share an edge with in the information graph. In a multi-agent system, having such a property is vital in making sure that each agent makes decisions in a decentralized manner, i.e., using only locally available information. We now formally define what it means for a function to be decentralized.

Definition 2.3. A function $\zeta : \mathcal{S} \times X^N \times \mathbb{R}^+ \to B^N$, where B is some arbitrary set, is *decentralized* if $\forall i \in \mathcal{N}$, $s \in \mathcal{S}$, $t \in \mathbb{R}^+$, and $x, \hat{x} \in X^N$, $\zeta^i(s, x, t) = \zeta^i(s, \hat{x}, t)$ when $x^j = \hat{x}^j \, \forall \, (i, j) \in E$, where $s(x) = (\mathcal{N}, E)$.

The function ζ is decentralized if ζ^i, its evaluation for agent i, is independent of the states of the agents that it does not share an edge with in the current information graph. Therefore, the computations that each agent performs in evaluating ζ^i use only *locally* available information.

Having defined what it means for a function to be decentralized, we can now insist that agents executing a controller in a consistent atom use only locally available information. This is done by ensuring that $\mathcal{U}$, the controller, and $\mathcal{C}$, the function used by agents to realize that the graph process has entered the set of final graphs, are both decentralized.

Definition 2.4. A *decentralized consistent atom* is a consistent atom $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$ where both $\mathcal{U}$ and $\mathcal{C}$ are decentralized.

2.3 Graph Process Specifications

Consistent atoms allow us to make guarantees on the information graph that results when the execution of its control law terminates. But, we have yet to define what determines the termination of a control law. One observation is that a control law within a consistent atom certainly cannot terminate until its information graph has entered the set of final graphs $\mathbb{H}$, as indicated by when $\mathcal{C}^i \to 1$, for some agent $i \in \mathcal{N}$. However, there may be additional termination conditions for the control law that are separate from those required for consistency. We let these additional conditions be represented by an "interrupt mapping" ξ.

For example, agents that are arranged in a line can be tasked to form a circle in at least three seconds. Assume that there exists a consistent atom whose control law drives a team of agents from a line to a circle formation. Let the interrupt mapping be such that $\xi^i \to 1$, for all $i \in \mathcal{N}$, for all time after the control law has executed for three seconds. The execution of the control law cannot end until the agents have entered the circle formation because subsequent tasks may require the agents to already be in a circle. Moreover, if the agents reach the circle formation in two seconds, the control law cannot terminate and move on to the next one in the sequence until at least three seconds has passed because we wish to follow the task specifications as closely as possible. Therefore, the termination condition for a control law within a consistent atom should be a logical AND of $\mathcal{C}^i$, the conditions required for consistency, and ξ^i, the interrupt mapping, as evaluated by an agent $i \in \mathcal{N}$. Notice that the decision to terminate the current control law is made by a single agent, who then needs to broadcast the termination command to all other agents in the network.

Agents executing a sequence of controllers can be viewed as a hybrid system, where a consistent atom $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$ and an interrupt mapping ξ determines a mode within the hybrid system, such that the agent dynamics within that mode evolve according to controller $\mathcal{U}$. The termination condition for the controller, or equivalently the guard condition for exiting the mode, is a logical AND of the function $\mathcal{C}^i$ and the interrupt mapping ξ^i, for some agent $i \in \mathcal{N}$. By further requiring that the consistent atom and interrupt mapping both be decentralized, the agents will be able to execute the mode from start to finish using only locally available information. With this formulation, we now formally define a mode:

Definition 2.5. A *mode* is denoted by the tuple

$$\mathcal{M} = (\mathcal{A}, \xi),$$

where $\mathcal{A}$ is a consistent atom and $\xi : \mathcal{S} \times X^N \times \mathbb{R}^+ \to \{0, 1\}^N$. Furthermore, a mode is a *decentralized mode* if $\mathcal{A}$ is a decentralized consistent atom and ξ is a decentralized function.

Observe that by keeping the consistency conditions $\mathcal{C}$ encapsulated within the atom, while letting the interrupt mapping ξ be specified separately when defining the mode, we allow consistent atoms to be reusable. For example, the same consistent atom $\mathcal{A}$ that shrinks a circle formation can be used to define two modes: one that terminates when the circle has radius smaller than 1, and another that terminates when the circle has radius smaller than 0.01, by simply using different interrupt mappings.

In our definition, a mode describes the execution of a single controller. Therefore, a complex task involving the execution of a sequence of controllers can be described by stringing together multiple modes. To provide a framework that describes the graph process resulting from a hybrid system executing such a sequence of modes, we define a Graph Process Specification (GPS). A GPS specifies a string of modes to be executed, along with a description of the agents' initial conditions, graph inducing function, and dynamics.

Definition 2.6. A *Graph Process Specification (GPS)* is a tuple given by

$$GPS = ((x_0, s^*, f^*), (\mathcal{M}_1, \ldots, \mathcal{M}_m)),$$

where $m \in \mathbb{N}$ and $\mathcal{M}_k = (\mathcal{A}_k, \xi_k)$, for $k = 1, \ldots, m$, such that

(1) $x_0 \in X^N$
(2) $s^* \in \mathcal{S}$
(3) $f^* \in \{f \in \mathcal{F} \mid f : X^N \times U^N \times \mathbb{R}^+ \to (TX)^N\}$.

To summarize the above definition, a GPS describes a hybrid system involving N agents by specifying the mode sequence to be executed. Furthermore, it describes the agents in the system where x_0 gives the initial state information, s^* is the graph inducing function that dictates how information flows amongst these agents, and f^* describes the agent dynamics. Note that a GPS is quite similar to a Motion Description Language (MDL), e.g., Brockett (1988).

2.4 Executable Graph Process Specifications

Although a GPS specifies a mode sequence, there is no guarantee that it is executable. We call two modes "composable" if no matter how the first mode terminates, the information graph always allows for the second mode to start. Composability requires that each element in the first mode's set of final graphs be a supergraph of some element in second mode's the set of initial graphs.

Definition 2.7. The mode $\mathcal{M} = (\mathcal{A}, \xi)$ is *composable* with the mode $\mathcal{M}' = (\mathcal{A}', \xi')$, where $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$ and $\mathcal{A}' = (\mathbb{S}', \mathbb{F}', \mathbb{G}', \mathbb{H}', \mathcal{U}', \mathcal{C}')$, if $\forall\, h \in \mathbb{H}$, $\exists\, g' \in \mathbb{G}'$ such that $g' \subseteq h$. We will denote this property by $\mathcal{M} \prec \mathcal{M}'$.

Note that mode composability does not necessarily commute. For example, a mode that drives agents from a line formation to a circle formation composes with a mode that rotates the circle formation, but not the other way around.

In order for a GPS to be executable, each pair of consecutive modes must be composable so that when agents terminate one mode of the sequence, they have an information flow graph rich enough to immediately begin executing the next mode. Since we are only concerned about the execution using a specific graph inducing function s^* and agent dynamics f^*, it is necessary to check that they fall into the sets $\mathbb{S}$ and $\mathbb{F}$, respectively, of each mode's consistent atom. Finally, the initial condition of the agents, x_0, needs to be such that the graph induced by it using s^* is rich enough to initiate the first mode in the sequence. If each mode in the GPS is also decentralized, then a team of agents can execute the GPS's complete mode sequence using only locally available information, with the exception of the global broadcasts used for mode switching. These requirements are described formally below:

Definition 2.8. A GPS consisting of m modes, given by $((x_0, s^*, f^*), (\mathcal{M}_1, \ldots, \mathcal{M}_m))$, is *executable* if

(1) $\mathcal{M}_k \prec \mathcal{M}_{k+1}$, for $k = 1, \ldots, m-1$
(2) $s^* \in \mathbb{S}_k$, for $k = 1, \ldots, m$
(3) $f^* \in \mathbb{F}_k$, for $k = 1, \ldots, m$
(4) $\exists g \in \mathbb{G}_1$ such that $g \subseteq s^*(x_0)$,

where $\mathcal{A}_k = (\mathbb{S}_k, \mathbb{F}_k, \mathbb{G}_k, \mathbb{H}_k, \mathcal{U}_k, \mathcal{C}_k)$, for $k = 1, \ldots, m$. Furthermore, an executable GPS is *locally executable* if each mode $\mathcal{M}_k$, $k = 1, \ldots, m$, is a decentralized mode.

2.5 Graph Process Specification Executions

Now that we know when a GPS is executable, we will formally describe what an execution of an executable GPS is. We start by defining a variant of the "hybrid time sets" used in Johansson et al. (1999) to describe the time intervals in which each mode of the GPS is being executed:

Definition 2.9. A *hybrid time set* is a sequence of intervals $Q = \{q_1, \ldots, q_w\}$, for some $w \in \mathbb{N}$, such that

(1) $q_k = [z_k, z'_k]$, for $k = 1, \ldots, w-1$
(2) $q_w = [z_w, z'_w]$ if $z'_w < \infty$, and $[z_w, \infty)$ otherwise
(3) $z_k \leq z'_k$, for $k = 1, \ldots, w$
(4) $z'_k = z_{k+1}$ for $k = 1, \ldots, w-1$.

Hybrid time sets are used to describe the execution of a GPS similar to how Johansson et al. (1999) uses them to describe the execution of a hybrid system: as a set of requirements on the state trajectory. A state trajectory is either accepted or rejected as an execution of the GPS.

To be an execution of a GPS, the state trajectory must begin at the initial condition specified in the GPS. The state evolution in each mode must be driven by the controller in that mode's consistent atom, as applied to the agent dynamics. Lastly, each mode terminates as soon as any agent detects that the information graph has entered into the set of final graphs and its interrupt mapping also fires. Although the end of a mode is detected by a single agent, all agents switch modes simultaneously.

Definition 2.10. An *execution* of an executable GPS, given of the form $((x_0, s^*, f^*), (\mathcal{M}_1, \ldots, \mathcal{M}_m))$, is a pair (Q, x), where $Q = \{q_1, \ldots, q_{\tilde{m}}\}$ is a hybrid time set with $\tilde{m} \leq m$ and $z_1 = 0$. If $\tilde{m} < m$, then $z'_{\tilde{m}} = \infty$, while if $\tilde{m} = m$, then we allow for $z'_{\tilde{m}} \leq \infty$. Additionally, $x(t)$ is a state trajectory defined on either $t \in [0, \infty)$ if $z'_{\tilde{m}} = \infty$, or on $t \in [0, z'_{\tilde{m}}]$ if $z'_{\tilde{m}} < \infty$, such that

(1) $x(0) = x_0$
(2) $\dot{x}(t) = f^*(x(t), \mathcal{U}_k(s^*, x(t), t), t)$ when $t \in q_k$, for $k = 1, \ldots, \tilde{m}$.
(3) For each $k = 1, \ldots, \tilde{m}-1$, $\exists i \in \mathcal{N}$ such that
$$\mathcal{C}_k^i(s^*, x(z'_k), z'_k) = 1 \text{ and } \xi_k^i(s^*, x(z'_k), z'_k) = 1.$$
If $z'_{\tilde{m}} < \infty$, then the above also holds for $k = \tilde{m}$.
(4) For each $k = 1, \ldots, \tilde{m}$, $\nexists\, t \in q_k - \{z'_k\}$ and $\nexists\, i \in \mathcal{N}$ such that
$$C_k^i(s^*, x(t), t) = 1 \text{ and } \xi_k^i(s^*, x(t), t) = 1.$$

This definition describes an execution of a graph specification in the following way: the state trajectory $x(t)$ of the agents starts at the initial condition x_0 at time $t = 0$. Given that the GPS contains a sequence of m modes, q_k corresponds to the time that mode k is being executed, for $k = 1, \ldots, \tilde{m}$ where $\tilde{m} \leq m$. In the kth mode, as indicated by when $t \in q_k$, the state dynamics of the agents f^* uses the controller $\mathcal{U}_k$ supplied by $\mathcal{A}_k$. The kth mode stops and switches to the $k+1$th mode in the sequence (or stops the execution of the GPS if $k = m$) the instant when both $C_k^i \to 1$ and $\xi_k^i \to 1$, for any agent $i \in \mathcal{N}$. Finally, since the end of the kth mode depends on a user defined interrupt mapping ξ_k, it is possible that the interrupt never fires, causing the kth mode to continue executing forever and never moving on to the $k+1$th mode (if there is one), which is why we allow $\tilde{m} \leq m$. Figure 1 provides an illustration showing how the execution of an executable GPS with three modes is viewed as a hybrid system.

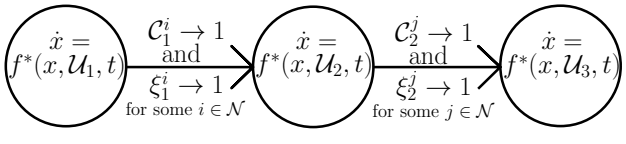


Fig. 1. An illustration showing the execution of an executable GPS with three modes as a hybrid system.

In the next section, we will give some examples of using GPS to design an executable sequence of controllers, and how GPS can be used to make an inexecutable sequence executable through mode insertions.

3. EXAMPLES OF GRAPH SPECIFICATIONS

In this section, we show how a GPS is used to determine whether a given sequence of controllers can be executed by a group of agents. More specifically, we provide an example of a locally executable GPS, which describes a sequence of controllers that drives N agents into a line formation, and then transitions into a circle formation. Each of the controllers used in the sequence will be encapsulated within a decentralized consistent atom. This section concludes by showing how an inexecutable GPS can help determine which additional modes can be inserted into its mode sequence in order to make it become executable.

Assume throughout the examples in this section that we have $N \geq 2$ agents, where the ith agent has state $x^i \in X = \mathbb{R}^2$ describing its planar position in Cartesian coordinates, for $i \in \mathcal{N}$. Since we are interested in focusing on coordination strategies at a high level, the agents are treated as point particles with dynamics given by a single integrator $f_{\mathcal{I}} : \mathbb{R}^{2N} \times \mathbb{R}^{2N} \times \mathbb{R}^{+} \to \mathbb{R}^{2N}$, where

$$f_{\mathcal{I}}(x, u, t) = u. \tag{1}$$

Furthermore, let

$$\mathbb{F}_{\mathcal{I}} = \{f_{\mathcal{I}}\} \tag{2}$$

be the set containing the single integrator dynamics.

We assume that an agent is able to communicate with another agent only if the distance between them is less than a constant $\delta > 0$. This assumption models situations where the agents communicate using wireless transceivers that are undirected and have a limited range. To describe such communication, let $s_\Delta(\delta) \in \mathcal{S}$ be the graph inducing function such that $\forall x \in \mathbb{R}^{2N}$, let $s_\Delta(\delta)(x) = (\mathcal{N}, E(x))$, with $(i, j) \in E(x) \Leftrightarrow ||x^i - x^j|| < \delta$. Furthermore, let

$$\mathbb{S}_\Delta(\delta) = \{s_\Delta(\delta)\} \tag{3}$$

be the set containing $s_\Delta(\delta)$, the Δ-disk graph inducing function with radius δ. Using these choices for agent dynamics and graph inducing functions, we now develop our example GPS.

3.1 Decentralized Consistent Atoms for Formation

We start by constructing decentralized consistent atoms $\mathcal{A}_{line}$ and $\mathcal{A}_{circ}$, that will drive N agents into a line and circle formation, respectively. In particular, the line formation in which we are interested in is the line graph $G_{line} = (\mathcal{N}, E_{line})$, where $E_{line} = \{(i, i+1) \,|\, i = 1, \ldots, N-1\}$. Let $\mathcal{U}_{line}(\delta)$ be any decentralized control law for the N agents taken from the diverse literature on decentralized formation control (e.g. Ji and Egerstedt (2007)) that will drive agents whose information graph is initially a supergraph of G_{line} to G_{line} in finite time. In other words, the set of initial and final graphs are identical, given by

$$\mathbb{G}_{line} = \mathbb{H}_{line} = \{G_{line}\}. \tag{4}$$

The circle formation we wish to achieve is given by the cycle graph $G_{circ} = (\mathcal{N}, E_{circ})$, where $E_{circ} = \{(N, 1)\} \cup \{(i, i+1) \,|\, i = 1, \ldots, N-1\}$. As with the line formation, have $\mathcal{U}_{circ}(\delta)$ be a decentralized control law that will drive agents whose information graph is initially a supergraph of G_{circ} to G_{circ} in finite time, where the initial and final graph sets are then given by

$$\mathbb{G}_{circ} = \mathbb{H}_{circ} = \{G_{circ}\}. \tag{5}$$

Finally, let $\mathcal{C}_{line}(\delta)$ and $\mathcal{C}_{circ}(\delta)$ be decentralized functions, whose implementation may vary depending on the choice of control law, that allows the agents to check locally whether the information graph has entered $\mathbb{H}_{line}$ and $\mathbb{H}_{circ}$, respectively. The decentralized consistent atoms for the line and circle formations are then given by:

$$\mathcal{A}_{line} = (\mathbb{S}_\Delta(\delta), \mathbb{F}_{\mathcal{I}}, \mathbb{G}_{line}, \mathbb{H}_{line}, \mathcal{U}_{line}(\delta), \mathcal{C}_{line}(\delta)) \tag{6}$$

$$\mathcal{A}_{circ} = (\mathbb{S}_\Delta(\delta), \mathbb{F}_{\mathcal{I}}, \mathbb{G}_{circ}, \mathbb{H}_{circ}, \mathcal{U}_{circ}(\delta), \mathcal{C}_{circ}(\delta)) \tag{7}$$

3.2 Locally Executable GPS Example

Now that we have designed decentralized consistent atoms that achieve line and circle formations, we can compose them to create a GPS. Let the initial conditions of the N agents, $x_0 \in \mathbb{R}^{2N}$, be chosen such that the induced graph of the agents' initial states is a complete graph, so that $G_{line} \subseteq s_\Delta(\delta)(x_0)$. To make things simple, we will let the interrupt mappings ξ_{line} and ξ_{circ} both be timer interrupts such that $\xi^i_{line} \to 1$ and $\xi^i_{circ} \to 1$, for all $i \in \mathcal{N}$ and for all time, after 3 seconds have elapsed since they were first evaluated. Combining these interrupts with the previously defined decentralized consistent atoms $\mathcal{A}_{line}$ and $\mathcal{A}_{circ}$, we can create the decentralized modes

$$\mathcal{M}_{line} = (\mathcal{A}_{line}, \xi_{line}) \text{ and } \mathcal{M}_{circ} = (\mathcal{A}_{circ}, \xi_{circ}), \tag{8}$$

which will achieve a line formation and a circle formation, respectively. These modes can be put into a sequence and used to create a GPS, which we will call GPS_1:

$$GPS_1 = ((x_0, s_\Delta(\delta), f_{\mathcal{I}}), (\mathcal{M}_{line}, \mathcal{M}_{circ})). \tag{9}$$

Note that this GPS is *not executable*. The problem that arises is that the modes $\mathcal{M}_{line}$ and $\mathcal{M}_{circ}$ are not composable since $G_{line} \in \mathbb{H}_{line}$ but $G_{circ} \notin G_{line}$ because there is no edge between agents 1 and N in G_{line}. However, this problem can be alleviated if we insert in a mode between $\mathcal{M}_{line}$ and $\mathcal{M}_{circ}$ that will drive agents with induced graphs in the set $\mathbb{H}_{line}$ to include an additional edge between agents 1 and N. To do so, we propose creating a new decentralized consistent atom that encapsulates the connectedness preserving nearest-neighbor averaging control law for dynamic graphs in Ji and Egerstedt (2007).

Theorem 1. Ji and Egerstedt (2007) state that if agents have the graph inducing function $s_\Delta(\delta)$, dynamics $f_{\mathcal{I}}$, and the induced graph is initially connected, then agents executing the decentralized control law $\mathcal{U}_{avg}(\delta)$, where

$$\mathcal{U}^i_{avg}(\delta)(s_\Delta(\delta), x, t) = -\sum_{j \in N(i)} \frac{2\delta - ||l_{ij}(x)||}{(\delta - ||l_{ij}(x)||)^2}(x^i - x^j), \tag{10}$$

for $i \in \mathcal{N}$, and $l_{ij}(x) = x^i - x^j$, will drive the agent positions to a single point asymptotically. Furthermore, the induced graph $s_\Delta(\delta)(x)$ will converge to K_N, the complete graph with N nodes, in finite time.

To detect that $s_\Delta(\delta)(x) = K_N$, the decentralized function $\mathcal{C}_{avg}(\delta)$ can be used, where for $i \in \mathcal{N}$:

$$\mathcal{C}^i_{avg}(\delta)(s, x, t) = \begin{cases} 1 \text{ , if } N(i) = \mathcal{N} - \{i\} \text{ and} \\ \quad ||x^i - x^j|| < \dfrac{\delta}{2} \,\forall\, j \in N(i) \\ 0 \text{ , otherwise.} \end{cases} \tag{11}$$

Letting $\mathbb{G}_{avg}$ be the set of all connected graphs and $\mathbb{H}_{avg} = \{K_N\}$, we can construct the decentralized consistent atom for nearest-neighbor averaging.

Lemma 2. The atom for performing nearest-neighbor averaging, $\mathcal{A}_{avg} = (\mathbb{S}_\Delta(\delta), \mathbb{F}_\mathcal{I}, \mathbb{G}_{avg}, \mathbb{H}_{avg}, \mathcal{U}_{avg}(\delta), \mathcal{C}_{avg}(\delta))$, is a decentralized consistent atom.

Proof. Theorem 1 states that when agents use the graph inducing function $s_\Delta(\delta)$ and control law $\mathcal{U}_{avg}(\delta)$, an induced graph that is originally in $\mathbb{G}_{avg}$ will converge to K_N in finite time and the agent positions will asymptotically approach a single point. When all agents are less than $\frac{\delta}{2}$ away from agent i, for some $i \in \mathcal{N}$, the graph must be complete and $\mathcal{C}^i_{avg} \to 1$.

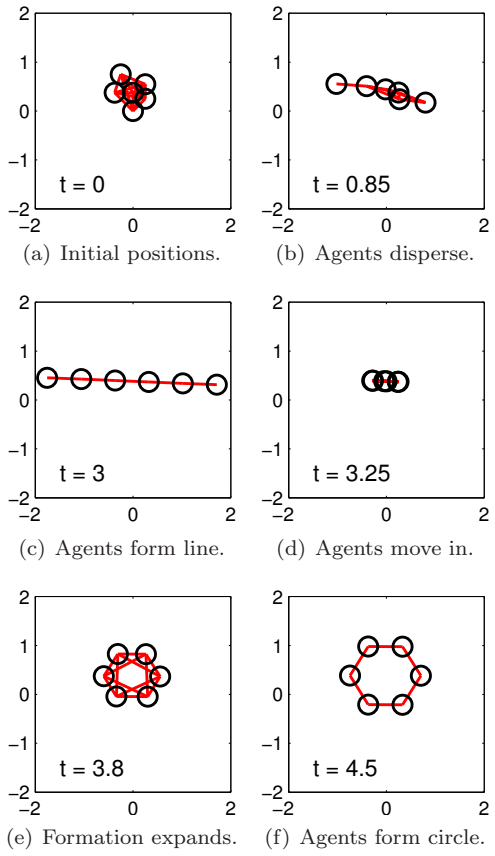
Define the mode

$$\mathcal{M}_{avg} = (\mathcal{A}_{avg}, \xi_{avg}), \tag{12}$$

where $\xi^i_{avg} \to 1$ always, for all $i \in \mathcal{N}$. We now construct a new GPS, which will be called GPS_2, given by

$$GPS_2 = ((x_0, s_\Delta(\delta), f_\mathcal{I}), (\mathcal{M}_{line}, \mathcal{M}_{avg}, \mathcal{M}_{circ})). \tag{13}$$

Since $G_{line} \in \mathbb{G}_{avg}$ and $G_{circ} \subseteq K_N$, we conclude that GPS_2 is locally executable. A simulation of agents executing GPS_2 is shown in Figure 2 for $N = 6$ and $\delta = 1$.



(a) Initial positions. (b) Agents disperse. (c) Agents form line. (d) Agents move in. (e) Formation expands. (f) Agents form circle.

Fig. 2. Simulation of GPS_2, given in (13), for $N = 6$ and $\delta = 1$. The location of the agents are marked by O's and the lines indicate edges in the induced graph.

The strategy of inserting $\mathcal{M}_{avg}$ into the inexecutable GPS_1 to form the executable GPS_2 is useful in many situations. As a consequence, $\mathcal{M}_{avg}$ can be thought of as an "universal glue" for modes that are not composable. We give the details of this observation in the theorem below.

Theorem 3. Let $\mathcal{M} = (\mathcal{A}, \xi)$ and $\mathcal{M}' = (\mathcal{A}', \xi')$ be two modes, where $\mathcal{A} = (\mathbb{S}, \mathbb{F}, \mathbb{G}, \mathbb{H}, \mathcal{U}, \mathcal{C})$ and $\mathbb{H} \subseteq \mathbb{G}_{avg}$. Furthermore, let $\hat{\mathcal{M}} = (\mathcal{A}_{avg}, \hat{\xi})$, where $\mathcal{A}_{avg}$ is defined in Lemma 2 and $\hat{\xi}$ is any arbitrary interrupt mapping. Then $\mathcal{M} \prec \hat{\mathcal{M}}$ and $\hat{\mathcal{M}} \prec \mathcal{M}'$.

Proof. $\mathcal{A}_{avg}$ is a consistent atom by Lemma 2 so $\hat{\mathcal{M}}$ is a mode. $\mathcal{M} \prec \hat{\mathcal{M}}$ because $\mathbb{H} \subseteq \mathbb{G}_{avg}$, and $\hat{\mathcal{M}} \prec \mathcal{M}'$ because K_N is a supergraph of any graph in $\mathcal{G}$.

4. CONCLUSION

In this paper we presented a Graph Process Specification (GPS) as a framework that describes the graph process associated with agents executing a sequence of controllers as a hybrid system, where each mode corresponded to the execution of a single controller. From the GPS, it is possible to determine whether a sequence of modes was executable by checking to see whether the information graph at the termination of one mode allowed for the controller in the next mode to execute. Furthermore, it was seen that a GPS also helps determine which modes can be inserted in order to make an inexecutable sequence executable. We concluded by presenting a detailed example of using a GPS to describe agents going into a line formation, and then transitioning to a circle formation.

REFERENCES

Brockett, R. (1988). On the computer control of movement. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, 534–540.

Ji, M. and Egerstedt, M. (2007). Distributed coordination control of multiagent systems while preserving connectedness. *IEEE Transactions on Robotics*, 23(4), 693–703.

Johansson, K., Egerstedt, M., Lygeros, J., and Sastry, S. (1999). On the regularization of Zeno hybrid automata. *Systems and Control Letters*, 38(3), 141–150.

Kloetzer, M. and Belta, C. (2007). Temporal logic planning and control of robotic swarms by hierarchical abstractions. *IEEE Transactions on Robotics*, 23(2), 320–330.

Manikonda, V., Krishnaprasad, P.S., and Hendler, J. (1998). Languages, behaviors, hybrid architectures and motion control. In J. Willems and J. Baillieul (eds.), *Mathematical Control Theory*. Springer-Verlag.

Martin, P., de la Croix, J., and Egerstedt, M. (2008). MDLn: A motion description language for networked systems. In *Proceedings of 47th IEEE Conference on Decision and Control*.

McNew, J. and Klavins, E. (2006). Locally interacting hybrid systems with embedded graph grammars. In *Decision and Control, 2006 45th IEEE Conference on*, 6080–6087.

Mesbahi, M. and Egerstedt, M. (2010). *Graph Theoretic Methods for Multiagent Networks*. Princeton University Press, Princeton, NJ.

Muhammad, A. and Egerstedt, M. (2005). Connectivity graphs as models of local interactions. *Applied Mathematics and Computation*, 168(1), 243–269.

Olfati-Saber, R., Fax, J., and Murray, R. (2007). Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1), 215–233.

Smith, B., Howard, A., McNew, J., Wang, J., and Egerstedt, M. (2009). Multi-robot deployment and coordination with Embedded Graph Grammars. *Autonomous Robots*, 26(1), 79–98.